Práctica 2.2: Formularios con Django

Objetivo

Crear un formulario web para insertar datos en la base de datos, utilizando Django Forms. Esto permitirá que un usuario cree nuevos registros del modelo Post desde una página web, no solo desde el admin.

- En esta práctica aprenderás a crear formularios web funcionales con Django para que los usuarios puedan insertar información directamente desde una página web, sin necesidad de usar el panel de administración.
- Esto es muy útil cuando quieres permitir que los visitantes o usuarios de tu sitio creen contenido (como artículos, comentarios, productos, etc.) desde la interfaz web.
- Para lograrlo, usaremos:
  - El módulo forms de Django para construir el formulario.
  - Vistas para manejar el envío del formulario.
  - Plantillas HTML para mostrar el formulario al usuario.
  - El modelo Post que ya hemos creado antes, con campos como título, contenido, y autor.

PASOS

Paso 1: Crear el entorno de trabajo.

1. En el explorador de archivos, dentro de la carpeta "CECYTEMDJangoPracticas" crea una carpeta llamada “practica2_2”.
2. Abre Visual Studio Code y selecciona File > Open Folder, abre la carpeta "practica2_2".
3. Crea un nuevo proyecto desde la terminal de VS Code “django-admin startproject blogformulario”. Esto crea una carpeta de publicaciones/.
4. Navegamos a nuestro proyecto “cd blogformulario”.

Paso 2: Crear una nueva app dentro del proyecto.

- Creamos desde la terminal de VS Code un nuevo proyecto llamado blog.

```
python manage.py startapp blog
```

Paso 3: Registrar la app en settings.py

- Abre blogformulario/settings.py y agrega 'blog', en INSTALLED_APPS:

```
INSTALLED_APPS = [
    ...
    'blog',
]
```

- Django necesita saber qué aplicaciones forman parte del proyecto para cargarlas correctamente.

Paso 4: Crear el modelo Post.

- Un modelo en Django es una representación en Python de una tabla de base de datos. Cada modelo se traduce en una tabla, y cada atributo de la clase es una columna de esa tabla. En blog/models.py:

```
from django.db import models

class Post(models.Model):
    titulo = models.CharField(max_length=200)
    contenido = models.TextField()
    autor = models.CharField(max_length=100)
    fecha_creacion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.titulo
```

- titulo: Es un campo de texto corto, limitado a 200 caracteres.
- contenido: Campo de texto largo, ideal para párrafos.
- autor: Nombre del autor como cadena, hasta 100 caracteres.
- fecha_creacion: Guarda la fecha y hora en que se crea el post.
- auto_now_add=True significa que Django lo llena automáticamente al crear el objeto.
- __str__: Este método le dice a Django cómo mostrar el objeto cuando lo veas en consola o en el panel de administración. Aquí, mostrará el título del post.

Paso 5: Aplicar migraciones (crear la tabla en la base de datos).

- Una vez creado el modelo, necesitamos decirle a Django que lo convierta en una tabla real dentro de la base de datos.

- Terminal VS Code:

python manage.py makemigrations

- Este comando crea un archivo de migración basado en el modelo. Es como una "instrucción" que Django leerá para modificar la base de datos.

python manage.py migrate

- Este comando aplica la migración y crea físicamente la tabla en la base de datos. Con esto, la tabla blog_post estará lista para guardar publicaciones.

Paso 6: Crear el formulario con forms.py

- Crea un archivo nuevo llamado forms.py dentro de tu app blog, y escribe:

```python
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['titulo', 'contenido', 'autor']
```

- Un formulario (Form) es una forma de interactuar con los datos en una página web (crear, editar, validar). Django permite crear formularios basados en modelos automáticamente, lo que facilita mucho el trabajo.

➡ forms.ModelForm: Django toma tu modelo y crea automáticamente campos del formulario con las mismas validaciones.

➡ Meta: Es una clase interna que configura el formulario.

- model: Le dices a Django qué modelo usar.

- fields: Especificas los campos del modelo que quieres incluir en el formulario.

Paso 7: Crear la vista para mostrar y procesar el formulario.

- Edita el archivo blog/views.py:

```python
from .forms import PostForm
from django.shortcuts import render, redirect

def crear_post(request):
    if request.method == 'POST':
        form = PostForm(request.POST) # Procesa los datos enviados
        if form.is_valid():
            form.save() # Guarda en la base de datos
            return redirect('post_exito') # Redirecciona a otra vista
    else:
        form = PostForm() # Muestra un formulario vacío
    return render(request, 'blog/crear_post.html', {'form': form})

def post_exito(request):
    return render(request, 'blog/post_exito.html')
```

- request.method == 'POST': Verifica si el formulario fue enviado.
- form.is_valid(): Valida los datos ingresados.
- form.save(): Guarda el objeto en la base de datos.
- redirect('post_exito'): Redirecciona a una página de éxito.

- Esto permite tanto mostrar el formulario como guardar los datos una vez enviados.

Paso 8: Configurar las URLs

- blog/urls.py (si no existe, créalo):

```python
from django.shortcuts import render, redirect
from .forms import PostForm

from django.urls import path
from . import views

urlpatterns = [
    path('crear/', views.crear_post, name='crear_post'),
    path('exito/', views.post_exito, name='post_exito'),
]
```

- ¿Qué hacen estas rutas?

- 'crear/': Muestra el formulario para crear un nuevo post.
- 'exito/': Muestra un mensaje o página de confirmación después de guardar el post.

- Enlazar estas URLs en el archivo principal del proyecto urls.py:

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('blog/', include('blog.urls')),  # Aquí se conecta la app blog
]
```

Paso 9: Crear las plantillas HTML

- Primero, asegúrate de tener un directorio llamado templates dentro de tu app blog. Ruta: blog/templates/blog/

- Crear el archivo: crear_post.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Crear Post</title>
</head>
<body>
    <h1>Crear un nuevo Post</h1>

    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Guardar</button>
    </form>
</body>
</html>
```

- {{ form.as_p }}: Renderiza el formulario como párrafos (<p>).

- {% csrf_token %}: Muy importante. Protege contra ataques CSRF (seguridad).

- Crear el archivo: post_exito.html

```
<!DOCTYPE html>
<html>
<head>
    <title>¡Éxito!</title>
</head>
<body>
    <h1>El post se ha creado correctamente 🎉</h1>
    <a href="/blog/crear/">Crear otro post</a>
</body>
</html>
```

Paso 10: Iniciar el servidor y entrar al panel de administración.

- En la terminal de VS Code escribimos el siguiente comando para correr nuestro programa:

  python manage.py runserver

- Entra en tu navegador a:

  http://127.0.0.1:8000/blog/crear/

- Verás el formulario para crear un nuevo post. Al enviarlo, si todo es válido, se guardará en la base de datos y redirigirá a: http://127.0.0.1:8000/blog/exito/